

- 1
- 2
- 3
- 4
- 5
- 6
- 7
- 8
- 9
- 10
- 11
- 12
- 13
- 14
- 15
- 16
- 17
- 18
- 19
- 20
- 21
- 22
- 23
- 24
- 25

2  
3  
4

## 6

7  
8  
9

## 11

12  
13  
14  
15  
16

17  
18  
19  
20  
21  
22  
23  
24

1       **Claim 14** recites a method of factoring operating system functions. A  
2 plurality of operating system functions that are used in connection with operating  
3 system resources are factored into first groups based upon first criteria. The first  
4 groups are also factored into individual sub-groups based upon second criteria.  
5 Each sub-group is assigned to its own programming object interface. A  
6 programming object interface represents a particular object's implementation of its  
7 collective methods that are effective to provide an object-oriented operating  
8 system. Individual objects having associated programming object interfaces are  
9 configured to be instantiated throughout a remote computing system.

10       The Office first asserts Orton at Col. 7, lines 1-67, Col. 8, Lines 1-14, and  
11 figure 4 for defining criteria that governs how functions of an operating system are  
12 to be factored and associating groups of functions with programming objects. The  
13 Applicant disagrees. The Office then correctly asserts that Orton "is not explicit  
14 with reference to the programming objects being configured to be instantiated  
15 throughout a remote computing system." *Office Action Dated March 11, 2004,*  
16 *Page 3.* The Applicant agrees, Orton does not disclose, teach or suggest  
17 "programming objects being configured to be instantiated throughout a remote  
18 computing system" as claimed in Claim 1.

19       The Office then asserts Chappell to correct the defects of Orton and asserts  
20 that "[o]ne would have been motivated to make such a modification to provide one  
21 application's services to another (page 4 lines 22 – 35)." *Office Action Dated*  
22 *March 11, 2004, Page 3.* The Applicant respectfully disagrees. It is respectfully  
23 submitted that the Office has engaged in impermissible hindsight reconstruction in  
24 rejecting the claims because none of the submitted references, alone or in  
25 combination, contain the motivation or suggestion for the asserted combination.

1 Additionally, the asserted combination of Orton and Chappell does not disclose,  
2 teach or suggest the methods of claims 1 and 14.

3 To establish a prima facie case of obviousness, three basic criteria must be  
4 met. First there must be some suggestion or motivation, either in the references  
5 themselves or in the knowledge generally available to one of ordinary skill in the  
6 art, to modify the reference or to combine reference teachings. Second, there must  
7 be a reasonable expectation of success. Finally, the prior art reference (or  
8 references when combined) must teach or suggest all the claim limitations. The  
9 teaching or suggestion to make the claimed combination and the reasonable  
10 expectation of success must both be found in the prior art and not based on  
11 applicant's disclosure. See *MPEP 706.02(j)* and *In re Vaeck*, 947 F.2d 488, 20  
12 USPQ2d 1438 (Fed. Cir. 1991). Thus, obviousness cannot be established by  
13 combining the teaching of the prior art to produce the claimed invention, absent  
14 some teaching or suggestion supporting the combination. The Office may not use  
15 the patent application as a basis for the motivation to combine or modify the prior  
16 art to arrive at the claimed invention.

17 Beginning at page 13 of the subject specification, an exemplary distribution  
18 is described of an operating system's resources across one process boundary and  
19 one machine boundary in a distributed computing system. In the described  
20 example, in relation to FIG. 4, resource object 48 is instantiated in-process (i.e.  
21 inside the application's process), resource object 50 is instantiated in another  
22 process on the same machine (i.e. local), and resource object 52 is instantiated on  
23 another machine (i.e. remote). Use of remote and local resources is described in  
24 an additional embodiment beginning at page 22 of the subject specification. An  
25 operating system is provided with the ability to determine, based on the specified

1 unique identifier, whether it has the resource that is requested. If it does not, the  
2 operating system can ascertain the location of the particular resource and retrieve  
3 it so that the application can have the requested resource. The location from  
4 which the resource is retrieved can be across process and machine boundaries. As  
5 an example, consider the following. If an application asks for a specific version of  
6 a "ReadFile" interface, and the operating system does not support that version, the  
7 operating system may know where to go in order to download the code to  
8 implement the requested functionality. Software code for the specific requested  
9 interface may, for example, be located on a web site to which the operating system  
10 has access. The operating system can then simply access the web site, download  
11 the code, and provide the resource to the application.

12         Orton is directed to an object-oriented interface for a procedural operating  
13 system in the "context of executing the object oriented application 103A on the  
14 computer platform 102." *Orton, Col. 8, Lines 21-22*. Orton describes a limited  
15 functionality procedural operating system, such as the Mach micro-kernel. *Orton,*  
16 *Col. 5, Line 67 to Col. 6., Line 2*. The executable entity in Mach is known as a  
17 thread. *Orton, Col. 11, Line 25*. Threads use ports to communicate with each  
18 other. A port is "basically a message queue inside the kernel if [sic] at threads can  
19 add messages to or remove message from, if they have the proper permissions to  
20 do so." *Orton, Col. 15, Lines 39-42*. Ports exist solely in the kernel and can only  
21 be manipulated via port rights. *Orton, Col 15, Lines 48-49*. Thus, Orton describes  
22 that messages are utilized by threads to communicate with each other through the  
23 use of ports. Nowhere in Orton is remote communication even mentioned. This is  
24 further supported by the Summary of the Invention in Orton, which is excerpted as  
25 follows:

1 The present invention is directed to a system and method of  
2 enabling an object-oriented application to access in an object-  
3 oriented manner a procedural operating system having a  
4 native procedure interface. ***The system includes a computer  
and a memory component in the computer.*** Orton, Col. 3,  
Lines 50-55 (*emphasis added*).

5 Therefore, the system described in Orton is implemented in a single computer.  
6 Indeed, nowhere in Orton is remote communication or a machine boundary even  
7 discussed. Rather, Orton describes a computer platform, such as “an International  
8 Business Machines (IBM) computer or an IBM-compatible computer .... [or an]  
9 Apple computer”. Orton, Col. 6, Lines 6-10.

10 The Office then asserts Chappell for disclosure of a remote computing  
11 system and as motivation for a combination of Chappell and Orton. Specifically,  
12 the Office asserts in the *Office Action Dated March 11, 2004, Page 3* that “[o]ne  
13 would have been motivated to make such a modification to provide one  
14 application’s services to another (page 4 lines 22 – 35).” Assuming that the Office  
15 is referring to page 4 of Chappell, the referenced portion of Chappell is excerpted  
16 as follows:

17 All OLE technologies and all the ActiveX technologies  
18 described in this book are built on the foundation provided by  
19 COM. So just what is COM? To answer this question, think  
20 first about another: how should on chunk of software access  
21 the service provided by another chunk of software? Today, as  
22 shown in Figure 1-2, the answer depends on what those  
23 chunks of software are. An application might, for example,  
24 link to a library and then access the library’s services by  
25 calling the function in the library. Or one application might  
use the service provided by another, which runs in an entirely  
separate process. In this case, the two local processes  
typically communicate by using an interprocess  
communication mechanism, which usually requires defining a  
*protocol* between the two applications (a set of message  
allowing one application to specify its request and the other to  
respond appropriately). A third example is an application that

1 might use a service provided by an operating system. Here  
2 the application commonly makes system calls, each of which  
3 is handled by the operating system. Or, finally, an application  
4 might need the services of software that is running on a  
5 completely different machine, accessible via a network.  
Many different approaches can be used to access these  
services, such as *exchanging messages with the remote  
application* or issuing remote procedure calls. *Chappell,  
Page 4, Line 23 to Page 5, Line 9 (emphasis added).*

6 As shown in the above excerpt, although Chappell describes that an application  
7 might need the services of software that is running on a completely different  
8 machine, Chappell does not include motivation or suggestion for associating  
9 groups of *operating system functions* with programming object being configured  
10 to be instantiated through a remote computing system. Rather, Chappell describes  
11 “exchanging messages *with the remote application*” and makes no mention of  
12 operating system functions associated with programming objects that are  
13 configured to be instantiated throughout a remote computing system. *Chappell,  
14 Page 5, Lines 8-9 (emphasis added).*

15 Chappell, for instance, states that “an application might need the services of  
16 software that is running on a completely different machine, accessible via a  
17 network”, but the referenced software is a remote application as shown in the  
18 above excerpt, e.g., “exchanging messages with *the* remote application”.  
19 *Chappell, Page 5, Line 5-9 (emphasis added).* Thus, the section asserted by the  
20 Office describes making one application’s services available to another  
21 application, and does not describe, teach or suggest *operating system functions*  
22 configured to be instantiated throughout a remote computing system. This is  
23 further supported in the assertion made by the Office, “[o]ne would have been  
24 motivated to make such a modification to provide *one application’s services to  
25 another* (page 4 lines 22 – 35).” *Office Action Dated March 11, 2004, Page 3*

1 (*emphasis added*). It is respectfully submitted that Chappell is relied on solely for  
2 support of a remote computing system, and does not include motivation or  
3 suggestion for being combined with Orton as asserted by the Office. Absent the  
4 present application, the Orton reference describes an object-oriented interface for a  
5 procedural operating system, while Chappell merely describes the use of COM  
6 objects for communication between applications. Even assuming solely for the  
7 sake of argument that Orton and Chappell are combinable as asserted by the  
8 Office, the combination does not disclose, teach or suggest the methods of claims  
9 1 and 14 as previously described. Accordingly, for at least these reasons, claims 1  
10 and 14 are allowable.

11 **Claims 2-13** depend either directly or indirectly from claim 1 and are  
12 allowable as depending from an allowable base claim. These claims are also  
13 allowable for their own recited features which, in combination with those recited  
14 in claim 1, are neither shown nor suggested in the references of record, either  
15 singly or in combination with one another. For example, claim 7 recites  
16 “instantiating a plurality of programming objects across a machine boundary”.  
17 Orton does not disclose, teach or suggest the presence of a machine boundary nor  
18 even mention more than one computer platform. Chappell does not cure the  
19 defects of Orton as previously described. Therefore, for at least this reason, claim  
20 7 is allowable.

21 **Claims 15-23** depend either directly or indirectly from claim 14 and are  
22 allowable as depending from an allowable base claim. These claims are also  
23 allowable for their own recited features which, in combination with those recited  
24 in claim 14, are neither shown nor suggested in the references of record, either  
25 singly or in combination with one another.

1       **Claim 29** recites an operating system application program interface  
2 embodied on a computer-readable medium comprising a plurality of object  
3 interfaces. Each object interface is recited to be associated with an object that  
4 includes one or more methods that are associated with and can call functions of an  
5 operating system that does not comprise the object interfaces. At least one object  
6 is configured to be remotely instantiated.

7       The Office correctly asserts that Orton “is not explicit with reference to at  
8 least one the objects being configured to be remotely instantiated.” *Office Action*  
9 *Dated March 11, 2004, Page 7*. The Office then asserts Chappell to correct the  
10 defects of Orton. As previously described, however, nowhere does Chappell  
11 disclose or even suggest any such subject matter. Chappell merely describes  
12 communication between applications, while Orton describes an operating system  
13 on a single computing device. It is respectfully submitted that the assertion of  
14 Chappell at Page 4, Lines 22-35 by the Office does not supply any suggestion or  
15 motivation for forming the asserted combination with Orton as previously  
16 described. As excerpted above, the referenced portion of Chappell merely  
17 describes exchanging messages with a remote application. Thus, Chappell does  
18 not correct the defects of Orton, alone or in combination. Chappell does not  
19 include suggestion or motivation for being combined with Orton nor does the  
20 combination disclose, teach, or suggest the operating system application  
21 programming interface of claim 29. Accordingly, for at least these reasons this  
22 claim is allowable.

23       **Claims 30-35** depend either directly or indirectly from claim 29 and are  
24 allowable as depending from an allowable base claim. These claims are also  
25 allowable for their own recited features which, in combination with those recited

1 in claim 29, are neither shown nor suggested in the references of record, either  
2 singly or in combination with one another.

3 **Claim 36** recites an operating system that includes a plurality of  
4 programming objects having interfaces. The programming objects represent  
5 operating system resources, and the interfaces define methods that are organized in  
6 accordance with whether they create an operating system resource or not. The  
7 programming objects are configured to be called either directly or indirectly by an  
8 application. The methods are configured to call operating system functions  
9 responsive to being called directly or indirectly by an application. The  
10 programming objects are configured to be instantiated throughout a remote  
11 computing system.

12 Again, the Office correctly asserts that Orton “is not explicit with reference  
13 to at least one the objects being configured to be remotely instantiated.” *Office*  
14 *Action Dated March 11, 2004, Page 8*. Orton neither discloses nor suggests  
15 “programming objects being configured to be instantiated throughout a remote  
16 computing system” as claimed in claim 36. As previously described, Chappell  
17 does not correct the defects of Orton. Chappell does not include suggestion or  
18 motivation for being combined with Orton nor does the combination disclose,  
19 teach, or suggest the operating system of claim 36. Accordingly, for at least this  
20 reason, claim 36 is allowable.

21 **Claims 37-40** depend from claim 36 and are allowable as depending from  
22 an allowable base claim. These claims are also allowable for their own recited  
23 features which, in combination with those recited in claim 36, are neither shown  
24 nor suggested in the references of record, either singly or in combination with one  
25 another.

1       **Claim 41** recites a method of converting an operating system from a non-  
2 object-oriented format to an object oriented formal. The operating system  
3 includes a plurality of operating system functions that are callable to create or use  
4 operating system resources. A plurality of programming object interfaces are  
5 defined that define methods that correspond to the operating system functions.  
6 Programming objects that support the interfaces are callable either directly by an  
7 application that makes object-oriented calls, or indirectly by an application that  
8 makes function calls. The programming objects being configured to be  
9 instantiated throughout a remote computing system. A programming object  
10 interface is called either directly via an object-oriented call, or indirectly via an  
11 indirection that transforms a function call into an object-oriented call. Responsive  
12 to the calling, an operating system function is called with a method of the  
13 programming object that supports said programming object interface.

14       The Office correctly asserts that Orton “is not explicit with reference to at  
15 least one the objects being configured to be remotely instantiated.” *Office Action*  
16 *Dated March 11, 2004, Page 9.* Orton neither discloses nor suggests  
17 “programming objects being configured to be instantiated throughout a remote  
18 computing system” as claimed in claim 41. Chappell does not correct the defects  
19 of Orton. Chappell describes communication with a remote application, and not  
20 programming object interfaces that define methods that correspond to the  
21 *operating system functions* and that are configured to be instantiated throughout a  
22 remote computing system. Accordingly, for at least these reasons, claim 41 is  
23 allowable.

1 For at least these reasons, claims 1-3, 6-17, 19, 22, 23, 29-31 and 36-41 are  
2 allowable over Orton in view of Chappell. Applicant respectfully requests that the  
3 §103 rejection of claims 1-3, 6-17, 19, 22, 23, 29-31 and 36-41 be withdrawn.

4  
5 **35 U.S.C. §103(a)**

6 Claims 4, 5, 18, 20, 21, 24-28 and 32-35 are rejected under 35 U.S.C.  
7 §103(a) as being unpatentable over Orton in view Chappell and further in view of  
8 U.S. Patent Number 6,334,157 to Oppermann et al. (hereinafter, "Oppermann").  
9 Applicant respectfully traverses the rejections.

10 **Claim 24** recites a method of factoring operating system functions which  
11 includes factoring a plurality of operating system functions into interface groups  
12 based upon the resources with which a function is associated. The interface  
13 groups are factored into interface sub-groups based upon each function's use of a  
14 handle that represents a resource. The interface sub-groups are organized so that  
15 at least one of the interface sub-groups inherits from at least one other of the  
16 interface sub-groups. Individual interface sub-groups are associated with  
17 individual programming objects that can be instantiated throughout a remote  
18 computing system.

19 The Office correctly asserts that Orton "is not explicit with reference to at  
20 least one the objects being configured to be remotely instantiated." *Office Action*  
21 *Dated March 11, 2004, page 12.* Neither Orton nor Chappell, alone or in  
22 combination, disclose or suggest a method that factors operating system functions  
23 where "individual objects sub-groups being associated with individual  
24 programming objects that can be instantiated throughout a remote computing  
25 system" as claimed in claim 24. Chappell does not correct the defects of Orton.

1 Additionally, Oppermann, alone or in combination with any of the submitted  
2 references, does not correct these defects.

3 The Office asserts Oppermann to describe factoring groups into interface  
4 sub-groups based upon each function's handle. In making out the rejection, the  
5 Office contends that Oppermann meets the subject matter of this claim.  
6 Specifically, the Office argues that Oppermann teaches a plurality of operating  
7 system functions, a handle, a resource, and organizing the interface sub-groups so  
8 that at least one of the interface sub-groups inherits from at least one other of the  
9 interface sub-groups. Applicant respectfully disagrees with the Office's  
10 interpretation of this reference and traverses the rejection.

11 Specifically, claim 24 recites a step in which the interface groups are  
12 factored into interface sub-groups *based upon each function's use of a handle*  
13 that represents a resource. Nowhere does Oppermann teach or even suggest this  
14 feature. The Office asserts Oppermann at Col. 8, Lines 16-67 and at Col. 21,  
15 Lines 35-51, which are excerpted as follows:

16 As part of the preferred architecture, an application program  
17 supports the IAccessible interface, which allows clients to  
18 reap the benefits of using the preferred architecture. As  
19 shown in FIG. 3, a client 302, such as an accessibility aid,  
20 accesses a server 304, such as an application program with  
21 user interface elements, via the IAccessible interface 306. The  
22 IAccessible interface 306 contains function members which  
23 allow the client 302 to directly access and manipulate the  
24 implementation of the user interface elements of the server  
25 304. These function members allow the client 302 to access  
the user interface elements in the following ways: by  
navigating through the user interface elements, by retrieving  
the name for each user interface element, by determining  
whether each user interface element is visible or hidden, by  
retrieving the text description for each user interface element,  
by identifying the location of the user interface element, by  
determining the parent or child of a user interface element,

1 and by determining the value of the user interface element.  
2 Some user interface elements have an associated value. For  
3 example, a clock user interface element has a value of the  
4 current time. *Oppermann, Col. 8, lines 16-67.*

#### 5 The LResultFromObject Function

6 The LResultFromObject function receives an identifier of an  
7 interface and returns an LResult. The LResultFromObject  
8 function is defined below.

9 LRESULT LresultFromObject(REFIID riid, WPARAM  
10 wParam, LPUNKNOWN punk);

11 This function receives a pointer to an IUnknown interface for  
12 a user interface element and generates a handle for the  
13 pointer, known as an LRESULT value. This function returns  
14 a positive number if successful or an error code otherwise.

15 riid--This parameter is a reference identifier of the interface  
16 that will be provided to the client. For example, the riid  
17 parameter may identify the IAccessible object interface.

18 wParam--This parameter is a value as provided in the wParam  
19 parameter received with the associated WM\_GetObject  
20 message. *Oppermann, Col. 21, lines 35-51.*

21 Thus, Oppermann simply describes the use of a handle. Oppermann neither  
22 discloses nor suggests factoring interface groups into interface sub-groups ***based***  
23 ***upon each function's use of a handle*** that represents a resource. Accordingly, for  
24 at least these reasons, this claim is allowable.

25 **Claims 25-28** depend either directly or indirectly from claim 24 and are  
allowable as depending from an allowable base claim. These claims are also  
allowable for their own recited features which, in combination with those recited  
in claim 24, are neither shown nor suggested in the references of record, either  
singly or in combination with one another.

**Claims 4, 5, 18, 20, 21 and 32-35** depend either directly or indirectly from  
allowable independent claims are allowable as depending from an allowable base

1 claim. These claims are also allowable for their own recited features. For  
2 example, claims 4 and 5 recite "creating a hierarchy of object interfaces in which  
3 certain interfaces can inherit from other interfaces" and "creating a hierarchy of  
4 object interfaces in which certain interfaces can aggregate with other interfaces",  
5 respectively, which are neither shown nor suggested in the references of record,  
6 either singly or in combination with one another.

7 For at least these reasons, claims 4, 5, 18, 20, 21, 24-28 and 32-35 are  
8 allowable over Orton in view of Chappell and further in view of Oppermann.  
9 Applicant respectfully requests that the §103 rejection of claims 4, 5, 18, 20, 21,  
10 24-28 and 32-35 be withdrawn.

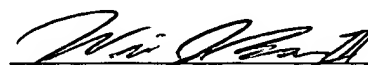
11  
12 **Conclusion**

13 All pending claims 1-41 are in condition for allowance. Applicant  
14 respectfully requests reconsideration and prompt issuance of the subject  
15 application. If any issues remain that prevent issuance of this application, the  
16 Examiner is urged to contact the undersigned attorney before issuing a subsequent  
17 Action.

18  
19 Respectfully Submitted,

20  
21 Dated: June 10, 2004

22 By:



23 William J. Breen, III  
24 Reg. No. 45,313  
25 (509) 324-9256 x249